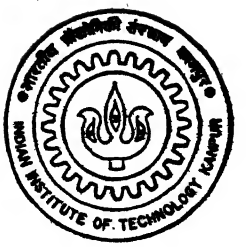
MULTIFONT DOCUMENT RECOGNITION

by

Sreekrishnan V.



DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

FEBRUARY, 1996

Multifont Document Recognition

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

Sreekrishnan V.

to the

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

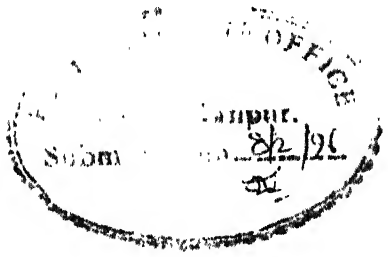
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

February 1996

CSE-1996-M-SRE-MUL

CERTIFICATE

This is to certify that the work contained in the thesis entitled Multifont Document Recognition by Sreekrishnan V. has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.



Dr. Harish Karnick,
Professor,
Department of Computer Science & Engineering,
Indian Institute of Technology, Kanpur.

February 1996.

Acknowledgements

I am grateful to Dr.Harish Karnick for his constant guidance throughout this work. The Class of '94 has made my stay here, a beautiful and memorable experience. I have enjoyed the wonderful company of my G-block wing mates and my friends in the hostel. I thank my parants and sister for giving me constant encouragement.

Abstract

This work attempts to build a complete prototype system that achieves multifont document recognition. A document image consists of multifont text and graphics. The first step is to separate the text regions from the graphics regions. Once this is done, Fourier analysis is done on the character segments to derive font independent features. To improve recognition rates, structural feature extraction methods are built over this. Word level post processing is now done to capture contextual information. Except for the structural and contextual processing, the system is independent of the script used in the document. For lossless document recognition, font recognition has to be done along with character recognition. All the information that have been thus captured from the document image have to be effectively represented in a text format. We use the HyperText Markup Language (HTML) as the text format, since it offers a compact set of symbols that can embed font information and graphics, in the recognized text. The system achieves over 95% recognition in many cases.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multifont document recognition aims to transform data addressed to human comprehension, into a form that can be recognized by computers. It achieves both character recognition and font recognition. In other words, it ideally achieves a lossless transformation of a document image into a text representation that can embed font information and graphics in the recognized text. Such a system can eliminate huge amounts of manual data entry work.

1.1 Objectives

The objective of this work is to transform a document image into a text form that is similar to an HTML document. Characters of different fonts have to be recognized along with font information like size, boldness etc. For the image areas in the document, hyperlinks to bitmap files containing the graphics portion are to be generated. This work proposes a complete prototype system that uses a collection of feature extraction methods to achieve multifont document recognition.

The first step in document recognition is to comprehend the layout and logical structure of the document. Here, the document is first separated into text and graphics areas. The text areas are further segmented into words, lines, sections etc. A degree of information is already captured at this stage, without attempting further processing, as in a check or a bill. To gain a further level of comprehension, the

characters have to be recognized. At this point, the system would have recognized the document as a typist would have done. But the typist might not have understood the language of the text he has typed. Use of word level processing with the aid of a language dictionary raises the level of recognition to that of a typist who understands the language words. Capturing the various font details increases comprehension to convey levels of emphasis or surprise. This work aims at this level of document recognition.

1.2 Thesis Overview

Chapter 2 describes the preprocessing steps involved. These include binarizing the document image and performing skew correction.
Chapter 3 describes the segmentation process. It deals with the separation of gra phics and character regions, generation of character boxes, line and word segmentation, and performing touching character segmentation.
Chapter 4 describes the Fourier analysis that is used to extract font independent shape features from a character contour. It explains the thinning algorithm used, the process of obtaining a character contour, generating the Fourier coefficients, properties of Fourier features, normalizing the Fourier coefficients for font independence, and classifying the obtained features .
Chapter 5 describes the structural features that were built above the Fourier features to boost the recognition rate. These include zero crossings, ascendents and descendents, position of the center of gravity, segment width etc.
Chapter 6 describes the contextual processing that was done to further improve the recognition rate. In particular, it deals with the spell check procedure and methods to guess the nearest correct word.
Chapter 7 describes the procedures used to capture font details. It also explains the conversion into HTML.
Chapter 8 describes the results obtained and performs error analysis.
Chapter 9 concludes the work and suggests future work that can be taken up.

Chapter 2

Preprocessing

Preprocessing refers to the operations that transform the image into a form acceptable to higher level modules. These operations include binarization, skew correction, image enhancement and other procedures depending on the specific application.

The image is typically obtained by scanning a paper-based document at an appropriate sampling resolution. Many scanners include the thresholding operation and can output a binary image. In our case, this was not done since the graphics area in the document had to be preserved while generating the image bitmaps. Simple noise reduction was performed during segmentation.

2.1 Binarization

The pixels in the foreground have to be extracted from those in the background. For this, a threshold T is selected to separate these classes of pixels. Let $I(x,y)$ represent the gray level image and let $I'(x,y)$ represent the bilevel image obtained after thresholding. Then any point in the image (x,y) for which $I(x,y) < T$, is decided as a foreground pixel, otherwise it is a background pixel.

$$I'(x,y) = 1 \; if \; I(x,y) > T,$$

$$I'(x,y) = 0 \; if \; I(x,y) \leq T.$$

Various algorithms exist for selecting the threshold. The PBMPLUS utilities were used to convert the input image into the PBM [KL92] format. This uses the

Floyd-Steinberg method [Uli] for thresholding. The threshold can also be configured by the user during run time.

2.2 Skew Correction

In document image analysis systems, printed material is scanned and stored as an image. During the scanning process, the text may be skewed and this may affect the higher modules. For instance, a tilted document may affect line segmentation.

In our system, we use the method proposed by Hong Yan [Yan93], for skew correction. This method determines the skew angle using cross-correlation between lines at a fixed distance. The correlation between two vertical lines of an image is maximized in general if one line is shifted relative to the other line such that the character page levels for the two lines are coincident. The accumulated correlation between pairs of lines separated by a constant distance is calculated for different shift values. Maximum correlation occurs when the shift corresponds to the skew of the image.

2.2.1 Skew Angle Detection

Let the image be represented by $B(x,y)$ where $0 \leq x \leq X-1$ and $0 \leq y \leq Y-1$. $B(x,y) = 0$ represents a foreground pixel and $B(x,y) = 1$ represents a background pixel. Consider the function

$$R(s) = \Sigma_{x=0}^{X-d-1} \Sigma_{y=s}^{Y-S-1} B(x+d, y+s) B(x,y)$$

where d is a constant and $-S \leq s \leq S$. s is the incremental shift to determine the correct tilt. S can be configured during run time, but a default value was empirically determined. $R(s)$ is the accumulated cross-correlation function for all pairs of lines at a fixed distance d in the image. Each term of the summation is 1 only if both (x,y) and $(x+d, y+s)$ are background pixels. $R(s)$ is maximum when s has the value S_{opt} for which there is maximum overlap of white spaces between text lines in the image.

The skew angle α can be determined from

$$\alpha = \arctan[\frac{s_{opt}}{d}]$$

The image is now rotated by α degrees in the opposite direction to correct the skew. The algorithm works even when there are graphics areas in the image. Also the multiplications were implemented as logical AND since in our case, the image is a bilevel one.

The output of this module is a binary image of the document that is in a form suitable for further processing. This is now sent to the segmentation module.

Chapter 3

Segmentation Of Graphics and Text

Segmentation is the process of isolating the constituent objects of an image. This is a step crucial to a successful solution. Broadly, our approach to segmentation is based on the discontinuity in pixel values, *i.e,* to partition the image based on abrupt changes in gray levels.

3.1 Separating Graphics and Text

A document image consists of multifont character regions and graphics regions. It can be viewed as a set of labeled connected components, each component being either a character segment or a graphics segment. The components are extracted by scanning each row of the image, from left to right. If a foreground pixel is encountered the entire component to which it belongs is recursively extracted and the start point of traversal noted. The components are labeled according to the order encountered. For a graphics segment, the area of the circumscribed rectangle will be high. The components thus identified are output as bitmap files. The pixel values prior to thresholding are output, to preserve the graphic regions unchanged. Segments for which the area of the circumscribed rectangle is inordinately low are rejected as noise. This enables the filtering of salt and pepper noise from the image.

The threshold below which the component is decided as noise, and the threshold above which it is considered as a graphics segment may depend on the particular image and these values can be configured during run time. All the remaining segments are treated as character boxes. The character boxes are now cleaned.This is necessary, since noise or neighbouring touching characters may be protruding into the character segment. Any foreground pixel in a character box, unreachable from the start point for the box, is erased. Figure 3.1 and Figure 3.2 illustrate the extraction of connected components.

3.2 Layout Analysis

For understanding a document, both the layout structure and the logical structure have to be extracted. This becomes evident when we consider checks, bills, letters etc. The layout specifies the appearance and is concerned with geometric properties. If the layout information is preserved, for instance, in a check, a human observer can associate categories of meaning like 'This is the amount' without the necessity of reading a single letter. Also, consider the case of a user accessing information over a network. He may just want to read the title of a certain paper. The layout of the paper may be presented to him and he can click on the portion desired and that part alone need be transmitted over the network. Thus layout analysis can be put to significant use even when recognition is not attempted subsequently and the document is stored only in an image form. [HBK92] discusses layout analysis and document understanding. Segmentation should, in addition to generating the connected components, preserve the layout structure of the document image. Different sections in the document image can be extracted based on paragraph separations and similar other characteristics. The distance between the connected components, their relative positions and size enable us to figure out the layout of the document image.

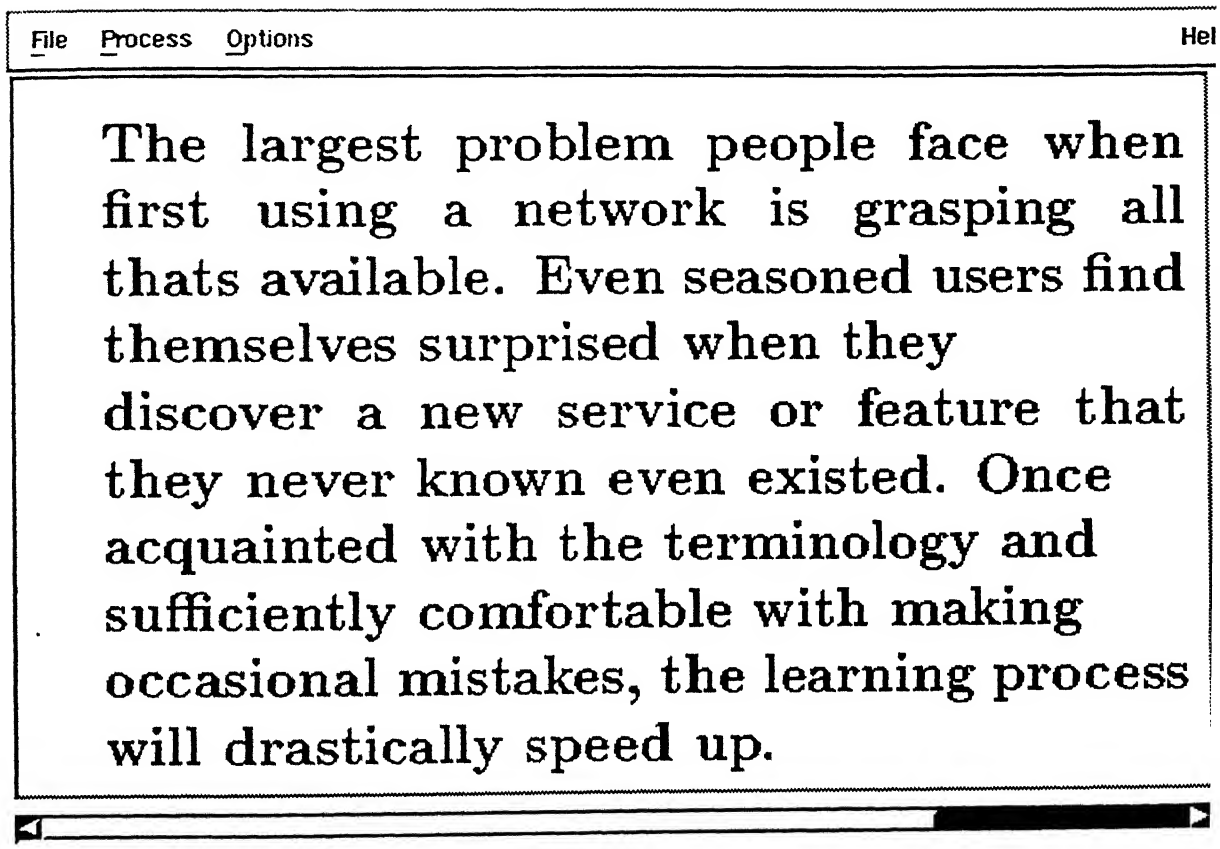


Figure 3.1 Input Image.

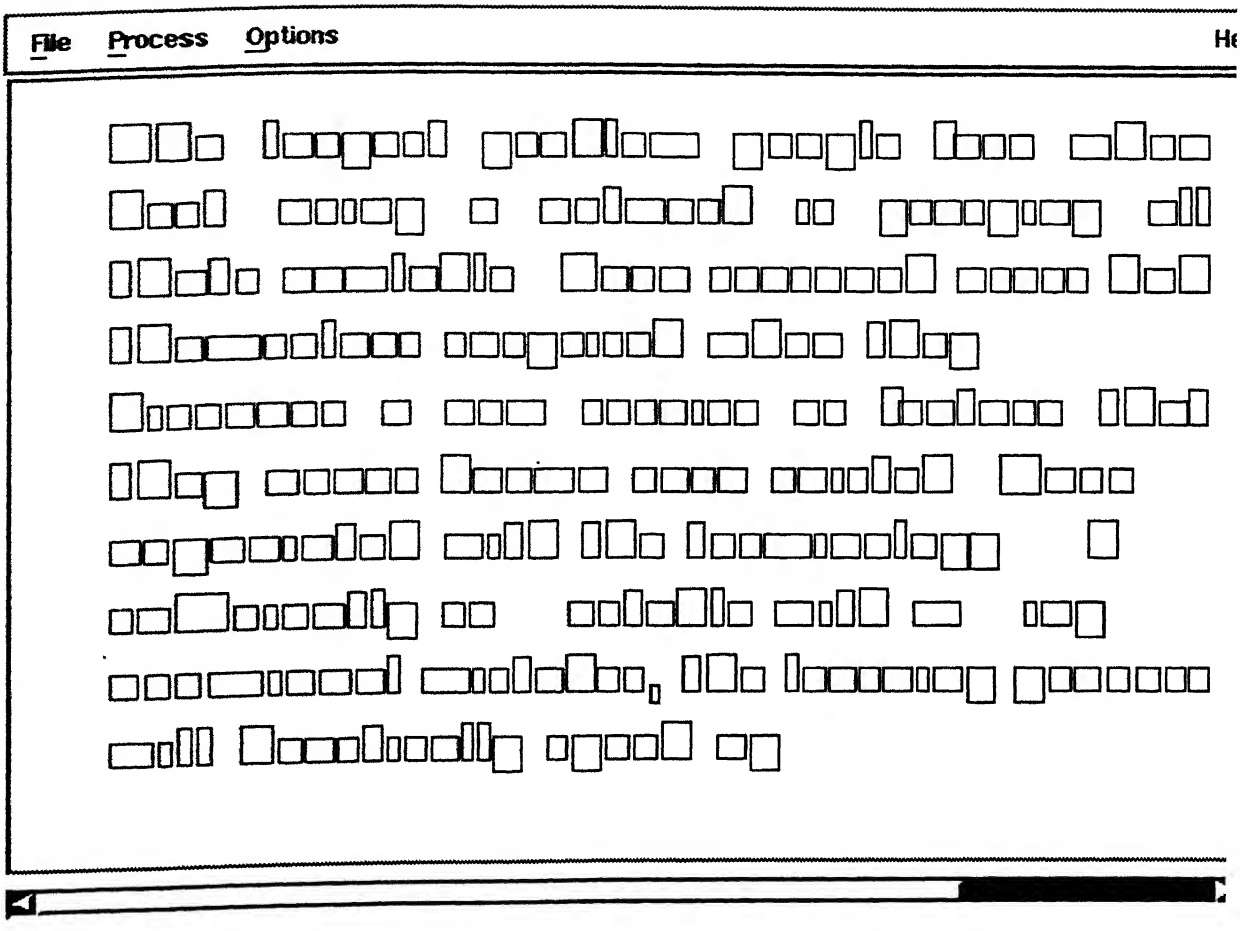


Figure 3.2 Connected Components in the Image.

3.3 Line Segmentation

Beginning of a new line can be detected by observing the difference in the top y-coordinate value of the current circumscribed box and the maximum bottom y-coordinate value of any character box encountered so far, in the current line. Once line segmentation is completed, the boxes in each line are sorted based on the x-coordinate value of the corners of each character box, to get the final correct labeling. The direction of the text lines were assumed to be horizontal, which may not hold always.

3.4 Word Segmentation

Word separation is performed whenever the top left x-coordinate of any character box differs from the top right x-coordinate of the previous box by a value greater than a fixed threshold. This threshold may depend on a particular image and hence can be configured during runtime.

3.5 Shadow Characters

Performing segmentation based on labeled connected components eliminate the problem of shadow characters. When a character resides in the shadow of another, the circumscribed rectangles of each character overlap. This can be seen in Figure 3.2 wherever shadow characters are present. But since each component is cleaned as previously described, those pixels in a character box that belong to a neighbouring shadow character are eliminated.

3.6 Touching Characters

Very often, in printed text, adjacent characters are touching and may exist in an overlapped fashion. We use the algorithm described in [LSM94] for segmenting touching characters. The algorithm uses two discrimination functions based on pixel

and profile projections. The pixel projection and profile projection are described as follows:

Pixel projection $PXP(k), k = 1.2 \ldots L$ is the total number of foreground pixels in a vertical column where L is the number of columns in the image.

Profile projection $PFP(k) = TP(k) - BP(k), k = 1, 2 \ldots L$ where $TP(k)$ is the top profile of the external contour of the touching character as seen from the top, *i.e*, the distance to the topmost foreground pixel in column k. $BP(k)$ is the bottom profile of the external contour of the touching character as seen from the bottom, *i.e*, the distance to the bottommost foreground pixel in column k.

The two segmentation discrimination functions are then defined as follows:

$$F_1^{\alpha}(k) = [\frac{PFP(k + L1) - 2PFP(k) + PFP(k - L1)}{PFP(k)}]$$

$$F_2^{\alpha}(k) = [\frac{PXP(k + L2) - 2PXP(k) + PXP(k - L2)}{PXP(k)}]$$

where $L1$ and $L2$ denote the distance between the current column and the adjacent column that are considered. These are empirically determined and depends on the size of the characters in the document. α is empirically determined as 2.

The final discrimination function used to compute the cutting points is defined as follows:

$$F(k) = \begin{cases} F_2^2(k) & \text{if } F_1^2(k) > T \text{ and } F_2^2(k) > T \\ F_1^2(k) & \text{if } F_1^2(k) > T \text{ and } F_2^2(k) < T \\ F_2^2(k) & \text{if } F_1^2(k) < T \text{ and } F_2^2(k) > T \\ 0 & \text{if } F_1^2(k) < T \text{ and } F_2^2(k) < T \end{cases}$$

This can be explained as follows. Consider two touching **o**'s. Here, $F_1^2(k)$ will have a large value at the touching point, while $F_2^2(k)$ will have a low value. In the case of a **t** touching an **h**, the opposite can be observed. Hence the final discrimination function should take the value of that segmentation function which crosses a threshold. The threshold values can be configured by the user during run time, but a default value was empirically determined.

High values of $F(k)$ determine candidate cutting points. The solution space thus obtained is exponential. Essentially, the procedure to be followed now is the

A^* algorithm [J.N82]. The theoretically exponential search space can be reduced using heuristics. No such heuristic was attempted in this work.

Touching characters present a major problem in document analysis. It is difficult to detect merged patterns in the text since the criterion to consider a connected component representing a touching character is based on the structural properties of the document and this varies from one font to another. For example, two touching **r**'s could be hard to detect due to the reasonable width of the resulting component. Hence, attempting touching character segmentation after classification, would yield better results. But even in this case, there are sources of potential misjudgements. For instance a touching **r** and an **n** may be classified as an **m**.

When the segmentation process is completed, the logical and layout structure of the document image would have been finalised. The graphics portion of the document image would have been output as bitmap files. The character components would have been structurally organized into words and lines. This is now input to the feature extraction module.

Chapter 4

Fourier Processing

The Fourier series aims to represent a periodic phenomena as a trigonometric series, *i.e,* as a sum of harmonically related sines and cosines. The concept of Fourier series and transforms have made a major impact on science and engineering. In this chapter, we discuss the application of the Fourier series to extract the shape features of a character contour. [OWY92] discusses Fourier analysis in detail.

To apply Fourier analysis on a character, the first step is to represent it as a closed contour of line segments. Tracing the boundary of the character is one method of contour representation. But this procedure is extremely sensitive to noise. The boundaries of character segments obtained after scanning the image was often found to be irregular and jarred. The contour thus obtained was found to significantly differ in shape depending on the document image and also on the font. Hence thinning is needed for good performance.

4.1 Thinning

The structural shape or skeleton of a character segment can be obtained by thinning. Thinning, in effect reduces the character segment into a skeleton graph. In this section, we discuss the thinning algorithm [GR93] that we used in our system.

The method basically consists of successive iterations of two basic steps applied to each foreground pixel in the character segment, which has at least one background

P9	P2	P3
P8	P1	P4
P7	P6	P5

Figure 4.1: Ordering of Adjacent Pixels

neighbour. Figure 4.1 shows the ordering of adjacent pixels

Step 1

Flag a contour point P for deletion if the following conditions are satisfied.

(a) $2 \leq N(P_1) \leq 6$

(b) $S(p_1) = 1$

(c) $P_2P_4P_6 = 0$

(d) $P_4P_6P_8 = 0$

where

$N(P_1) = P_2 + P_3 + \ldots + P_8 + P_9$, the number of nonzero neighbours of P_1.

and

$S(P_1)$ is the number of $0-1$ transitions in the sequence $P_2, P_3 \ldots P_9, P_2$.

Step 2

Steps (a) and (b) remain the same.

(c') $P_2P_4P_8 = 0$

(d') $P_2P_6P_8 = 0$

Algorithm

repeat

*Apply step*1 *to all border pixels.*

Flag a pixel for deletion if all conditions are satisfied.

After all pixels have been processed, delete all pixels flagged for deletion.

Apply step2 to the resulting data in a similar manner.

until (no more pixels can be deleted)

Step(a) prevents dismembering the skeleton and erosion into a segment. step(b) prevents disconnection of a segment. Steps (c) and (d) along with the earlier steps require that a point should be an east or south or northwest corner point to be flagged for deletion. Similarly (c') and (d') require that a point should be north or west or a southeast corner point to be flagged for deletion.

This algorithm results in a reasonably good reduction of a character segment into a skeleton graph form. Figure 4.2 illustrates this.

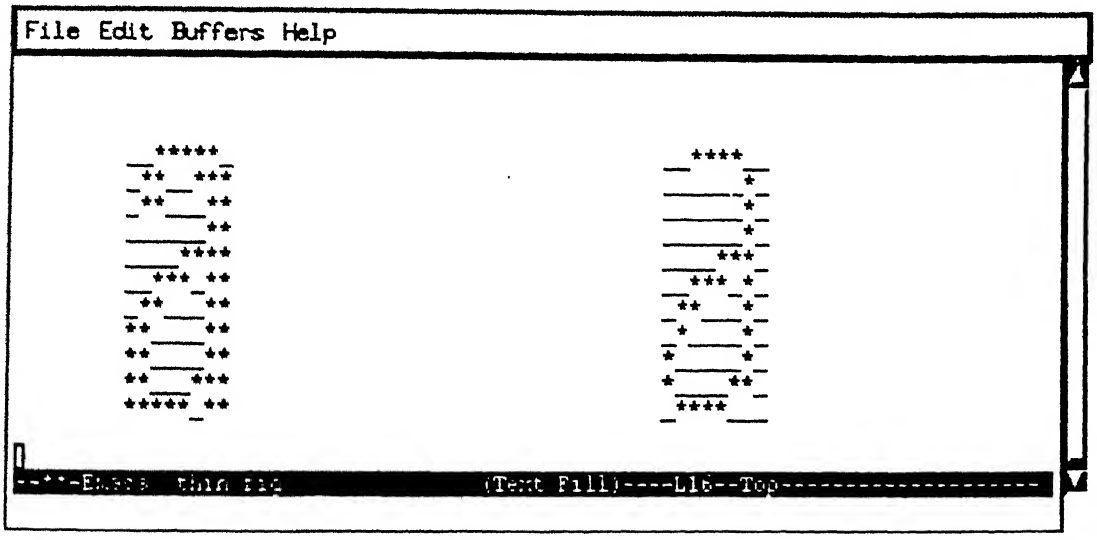


Figure 4.2 A Character Segment before and After Thinning

The thinned segment is input to the Fourier analysis module for Fourier feature extraction.

4.2 Obtaining the Contour

The thinned character segment is now traced to obtain a closed contour that is suitable for Fourier analysis. Traversal is done by starting at a point and hopping to an adjacent pixel that is still untraversed. Each traversed pixel is flagged as

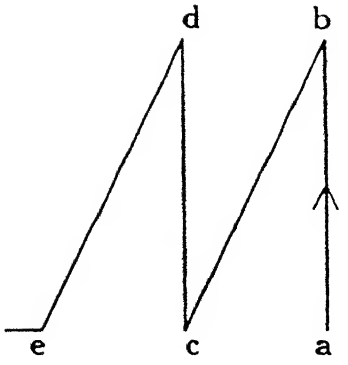


Figure 4.3: Traversal Before Smoothening

traversed and pushed into a stack. When a point is reached where there are no more adjacent untraversed pixels, the path is retraced by popping pixels from the stack. Popping is done till we reach a point where there are untraversed adjacent pixels. This process is continued till we reach back the starting pixel. The starting point is chosen such that it is not part of a closed curve in which case the shape of the contour itself changes. This need not be done for those characters which are made up of only closed curves.

Further, a smooth contour can be generated by following a simple heuristic. Choose pixels that are horizontally or vertically adjacent before pixels that are diagonally adjacent. It was found that this heuristic effects significantly better performance. Figure 4.3 and Figure 4.4 illustrate this.

4.3 Generating Fourier Coefficients

While traversing the contour as described in the previous section, the changes in the x, y projections and the time taken are continuously computed. Assuming the origin as the top left corner of the thinned character box, the x, y projections at any step i are,

$\Delta x_i = x_{i+1} - x_i,$

$\Delta y_i = y_{i+1} - y_i,$

where $\Delta x_i, \Delta y_i \epsilon \{1, 0, -1\}$ depending on the position of the adjacent pixel that is

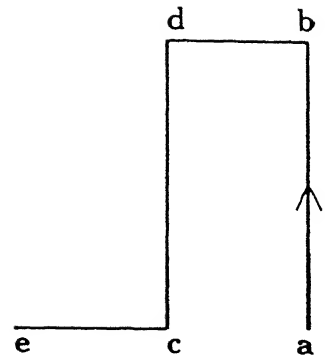


Figure 4.4: Traversal After Smoothening

chosen to be traversed next.

The time required for any hop is $\sqrt{2}$ units if the next pixel is a diagonally adjacent one and 1 unit otherwise. This is because the ratio of the length of the diagonal of a square to a side is $\sqrt{2} : 1$.

$\Delta t_i = t_{i+1} - t_i \epsilon \{1, \sqrt{2}\}$.

Also the time period

$$T = \Sigma_{i=1}^{k} \Delta t_i,$$

where k is the number of hops done during contour traversal. From this point, we use the method described in [PG82].

The Fourier series expansion for the x projection of a closed contour is defined as

$$x(t) = A_0 + \Sigma_{n=1}^{\infty} a_n cos\frac{2n\pi}{T}t + b_n sin\frac{2n\pi}{T}t,$$

where

$A_0 = \frac{1}{T}\int_0^T x(t)dt$,

$a_n = \frac{2}{T}\int_0^T x(t)\cos\frac{2n\pi}{T}tdt$ and

$b_n = \frac{2}{T}\int_0^T x(t)\sin\frac{2n\pi}{T}tdt$.

The time derivative is periodic with period T and can itself be represented by the Fourier series

$$\dot{x}(t) = \Sigma_{n=1}^{\infty} \alpha_n \cos\frac{2n\pi}{T}t + \beta_n \sin\frac{2n\pi}{T}t,$$

where

$\alpha_n = \frac{2}{T}\int_0^T \dot{x}(t)\cos\frac{2n\pi}{T}tdt$ and

$\beta_n = \frac{2}{T}\int_0^T \dot{x}(t)\sin\frac{2n\pi}{T}tdt.$

Then,

$$\alpha_n = \frac{2}{T}\Sigma_{p=1}^{k}\frac{\Delta x_p}{\Delta t_p}\int_{t_{p-1}}^{t_p}\cos\frac{2n\pi}{T}tdt$$
$$= \frac{2}{T}\Sigma_{p=1}^{k}\frac{\Delta x_p}{\Delta t_p}(\sin\frac{2n\pi}{T}t - \sin\frac{2n\pi}{T}t_{p-1}) \text{ and}$$
$$\beta_n = \frac{2}{T}\Sigma_{p=1}^{k}\frac{\Delta xp}{\Delta t_p}\int_{t_{p-1}}^{t_p}\sin\frac{2n\pi}{T}tdt$$
$$= \frac{-2}{T}\Sigma_{p=1}^{k}\frac{\Delta x_p}{\Delta t_p}(\cos\frac{2n\pi}{T}t_p - \cos\frac{2n\pi}{T}t_{p-1}).$$

But $\dot{x}(t)$ can also be obtained directly from it's definition as:

$$\dot{x}(t) = \Sigma_{n=1}^{\infty}\frac{-2n\pi}{T}\alpha_n\sin\frac{2n\pi}{T}t + \frac{2n\pi}{T}b_n\cos\frac{2n\pi}{T}t.$$

Equating coefficients from both these equations of $\dot{x}(t)$, we get,

$$a_n = \frac{T}{2n^2\pi^2}\Sigma_{p=1}^{k}\frac{\Delta x_p}{\Delta t_p}[\cos\frac{2n\pi}{T}t_p - \cos\frac{2n\pi}{T}t_{p-1}],$$
$$b_n = \frac{T}{2n^2\pi^2}\Sigma_{p=1}^{k}\frac{\Delta x_p}{\Delta t_p}[sin\frac{2n\pi}{T}t_p - sin\frac{2n\pi}{T}t_{p-1}].$$

Similarly the Fourier series expansion for the y projection can be found out as:

$$y(t) = C_0 + \Sigma_{n=1}^{\infty}c_n\cos\frac{2n\pi}{T}t + d_n\sin\frac{2n\pi}{T}t,$$

where

$$c_n = \frac{T}{2n^2\pi^2}\Sigma_{p=1}^{k}\frac{\Delta y_p}{\Delta t_p}[\cos\frac{2n\pi}{T}t_p - \cos\frac{2n\pi}{T}t_{p-1}],$$
$$d_n = \frac{T}{2n^2\pi^2}\Sigma_{p=1}^{k}\frac{\Delta y_p}{\Delta t_p}[\sin\frac{2n\pi}{T}t_p - \sin\frac{2n\pi}{T}t_{p-1}].$$

The derivation of the dc components A_0 and C_0 are not relevant here and hence not discussed.

4.4 Properties of Fourier Features

The truncated Fourier approximation to a closed contour can be written as,

$$x(t) = A_0 + \Sigma_{n=1}^{N}X_n,$$
$$y(t) = C_0 + \Sigma_{n=1}^{N}Y_n,$$

where

$X_n(t) = a_n\cos\frac{2n\pi}{T}t + b_n\sin\frac{2n\pi}{T}t$ and

$Y_n(t) = c_n\cos\frac{2n\pi}{T}t + d_n\sin\frac{2n\pi}{T}t.$

and all the (X_n, Y_n) pairs have an elliptic loci. The contour can be considered as the phasor sum of the elliptic loci. This is illustrated in Figure 4.5 [PG82].

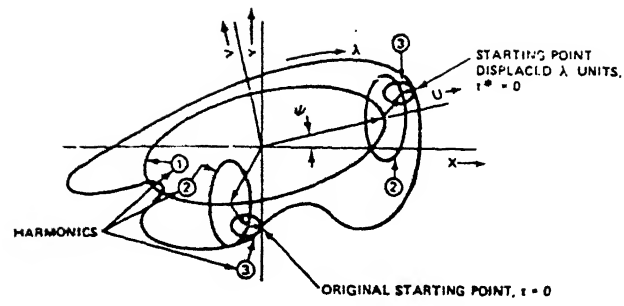


Figure 4.5 Elliptic representation of a contour

A difference in starting point does not affect the elliptic loci. Only the phasor relationship changes, *i.e*, a phasor shift is introduced in the calculations. That is,

$$\begin{aligned} X_n(t^* + \lambda) &= a_n \cos \tfrac{2n\pi}{T}(t^* + \lambda) + b_n \sin \tfrac{2n\pi}{T}(t^* + \lambda) \\ &= a_n^* \cos \tfrac{2n\pi}{T} t^* + b_n \sin \tfrac{2n\pi}{T} t^* \text{ and} \\ Y_n(t^* + \lambda) &= c_n \cos \tfrac{2n\pi}{T}(t^* + \lambda) + d_n \sin \tfrac{2n\pi}{T}(t^* + \lambda) \\ &= c_n^* \cos \tfrac{2n\pi}{T} t^* + d_n \sin \tfrac{2n\pi}{T} t^*, \end{aligned}$$

where

$$\begin{bmatrix} a_n^* & c_n^* \\ b_n^* & d_n^* \end{bmatrix} = \begin{bmatrix} \cos \frac{2n\pi}{T} t & \sin \frac{2n\pi}{T} t \\ -\sin \frac{2n\pi}{T} t & \cos \frac{2n\pi}{T} t \end{bmatrix} \begin{bmatrix} a_n & c_n \\ b_n & d_n \end{bmatrix}$$

Counter clockwise rotation of the $x - y$ axes by ψ degrees affects the $U - V$ axes as follows:

$$\begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

The new coefficients can be expressed as

$$\begin{bmatrix} a_n^{**} & c_n^{**} \\ b_n^{**} & d_n^{**} \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} a_n & b_n \\ c_n & d_n \end{bmatrix}$$

The length of the semimajor axis of the first harmonic ellipse directly corresponds to the size of the character contour.

4.5 Elliptic Features

The Fourier coefficients derived above are not independent of size, rotation, starting point etc. The following section deals with the normalization procedures that make these features independent of the above font dependent characteristics.

When the locus of the first harmonic of the Fourier series is elliptical, efficient normalization procedures are available. When the locus is circular, the computations involved are very extensive. But in character recognition, the latter case rarely arise. A circular loci is obtained when there is perfect symmetry in the character segment. The only possibility in our domain is o, which can be recognized by just this property. Noise rarely introduces perfect symmetry, and in this case, even if that happens, the character would have become so distorted that correct recognition is anyway impossible. Hence we safely eliminated the circular locus case from [PG82].

4.5.1 Elliptical First Harmonic Locus

To make the analysis independent of starting point, we rotate the first harmonic phasor till it coincides with the semimajor axis of the ellipse. This gives the effect of calculating the features from a point in the contour corresponding to the semimajor axis of the ellipse. To make the analysis independent of rotation, the $X-Y$ axes are rotated into a new axis coinciding with the semimajor and semiminor axis. To make the analysis independent of size, we scale the Fourier coefficients by the length of the semimajor axis of the first harmonic ellipse.

Two sets of features exist due to the presence of two semimajor axes. If θ_1 denotes the starting point rotation and ψ_1 denotes the spatial angular rotation, the normalized features are given by the following equations:

$$\begin{bmatrix} {}_1a_n^{**} & {}_1b_n^{**} \\ {}_1c_n^{**} & {}_1d_n^{**} \end{bmatrix} = \begin{bmatrix} \cos\psi_1 & \sin\psi_1 \\ -\sin\psi_1 & \cos\psi_1 \end{bmatrix} \begin{bmatrix} a_n & b_n \\ c_n & d_n \end{bmatrix} \begin{bmatrix} \cos n\theta_1 & -\sin n\theta_1 \\ \sin n\theta_1 & \cos n\theta_1 \end{bmatrix}$$

$$\begin{bmatrix} {}_2a_n^{**} & {}_2c_n^{**} \\ {}_2b_n^{**} & {}_2d_n^{**} \end{bmatrix} = (-1)^{n+1} \begin{bmatrix} {}_1a_n^{**} & {}_1b_n^{**} \\ {}_1c_n^{**} & {}_1d_n^{**} \end{bmatrix}$$

The starting point rotation θ_1, is determined as follows:

$x_1 = a_1 \cos\theta + b_1 \sin\theta,$

$y_1 = c_1 \cos\theta + d_1 \sin\theta.$

Differentiating and setting the derivative to zero, we get:

$$\theta_1 = \frac{1}{2} \arctan\left[\frac{2(a_1b_1 + c_1d_1)}{a_1^2 + c_1^2 - b_1^2 - d_1^2}\right]$$

This may find any of the semimajor or semiminor axes, depending on whether we arrive at a maxima or a minima. This can be clarified by comparing the axis length with that of the semimajor axis of the first harmonic ellipse. If a minima is encountered, a further addition of 90 degrees is done to θ_1. Here [PG82] stands corrected.

The spatial rotation ψ_1 is determined from the coefficients that are already corrected for starting point displacement. $(i.e, a_1^*, b_1^*, c_1^*, d_1^*)$

$$\begin{bmatrix} a_1^* & c_1^* \\ b_1^* & d_1^* \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} a_1 & c_1 \\ b_1 & d_1 \end{bmatrix}$$

$x_1^*(t) = a_1^* \cos\frac{2\pi}{T}t + b_1^* \sin\frac{2\pi}{T}t$
$y_1^*(t) = c_1^* \cos\frac{2\pi}{T}t + d_1^* \sin\frac{2\pi}{T}t$

$$\psi_1 = \arctan\left[\frac{y_1^*(0)}{x_1^*(0)}\right] = \arctan\left[\frac{c_1^*}{a_1^*}\right], 0 \leq \psi_1 \leq 2\pi.$$

Further, the length of the semimajor axis of the first harmonic ellipse is given by $(a_1^{*2} + c_1^{*2})^{\frac{1}{2}}$. Scaling the Fourier coefficients by this value makes the features size independent.

4.6 Classification

During training runs, the Fourier features are calculated on the sample segments, and stored. Dynamic learning was not done. The first five harmonics were used for classification decisions. A recognition decision is based on a metric D_m, between a known class m and the unknown class where D_m is defined as:

$$D_m^2 = (min)_{r=1,2...R}[(min)_{p=1,2...P}\Sigma_{n=1}^{N} D_n^2(r,p,m)].$$

where

$$D_n^2(r,p,m) = ({}_r a_n^{**} -_p a_{mn}^{**})^2 + ({}_r b_n^{**} -_p b_{mn}^{**})^2 + ({}_r c_n^{**} -_p c_{mn}^{**})^2 + ({}_r d_n^{**} -_p d_{mn}^{**})^2.$$

where m stands for the class number,r,p stand for the feature numbers and n stands for the harmonic number. R and P stand for the number of F[redacted] features

computed for the unknown and known classes respectively. In our case $R = P = 2$. This metric is determined for the unknown contour and all the known classes. The minimum of these unknown metrics determine the class of the unknown character contour.

Fourier analysis can be used irrespective of the script used in the document. A common problem encountered in scientific documents is the processing of mathematical expressions. Since Fourier analysis captures the shape of any character contour, it can also generate feature values for mathematical symbols. The equivalent text form could be represented in TEX. This can be done as an extension to this work.

The Fourier analysis method along with all it's advantages, and properties of font independence, does not result in acceptable recognition rates. Properties like rotation independence, while contributing to font independence on the one hand, result in certain wrong decisions on the other. For instance, **c**, **n** and **u** will have Fourier descriptors that are close to each other due to the rotation invariance property of the normalized Fourier descriptors.To get acceptable recognition rates, we use structural features in addition to the Fourier features. The next chapter describes the structural features that were used.

Chapter 5

Structural Features

The Fourier features deal with the shape in a quantitative fashion and largely ignores any structural property inherent in the pattern. For this reason, when encountering character patterns of different fonts, Fourier features may tend to confuse a character with another that is close to it in shape. Table 5.1 lists the mismatches that were most frequent. The system so far, with minor modifications, can function as a recognizer for any script. This chapter and the subsequent one, deals with features that are peculiar to the Roman script. For a different script, these features have to be replaced by equivalent script specific structural and semantic features.

Hence we added additional features that take cognizance of the structural peculiarities of a character segment. These features improved the recognition rate

n	c	u
m	w	
u	v	
a	d	
i	j	l
h	b	
o	n	
a	q	
n	h	

Table 5.1: List of Ambiguous Characters

i	l	j
n	h	p
a	d	q

Table 5.2: Characters Disambiguated Using Ascenders and Descenders

significantly. The additional feature extraction methods that we used are explained below. All these features are computed on the raw character segment, *i.e*, the segment obtained prior to thinning.

5.1 Ascenders and Descenders

In the English language, characters can be written within four guiding lines. A character has an ascender if it occupies space between the top two guide lines. It is said to have a descender if it occupies space between the bottom two guide lines. A character was decided as possessing an ascender if the top y-coordinate of the character box was less than the average that was computed for that line. Similarly, it was decided as possessing a descender if the bottom y-coordinate was greater than the average for that line. This feature disambiguates certain sets of characters that the Fourier features often render ambiguous. Table 5.2 lists the characters that could be disambiguated by this feature.

5.2 Zero Crossing

Zero crossing is the number of $0-1$ transitions in a line in an image. This feature enabled disambiguation of several pairs of characters that were close in terms of Fourier features. For instance, u and v are close to each other in shape except for the twist present in the right side of u. Moreover thinning sometimes erases this twist. But it can be observed that computing the zero crossing for the bottom horizontal line, in the raw image yeilds a value of 2 for u and 1 for v. Thus zero crossings can disambiguate u and v. Table 5.3 lists the pairs of characters that were

u	v
c	u
b	h
e	c
o	n

Table 5.3: Characters Disambiguated using Zero Crossings

m	w
n	u
6	9

Table 5.4: Characters Disambiguated Using Position of CG

disambiguated successfully, by considering zero crossings in different directions, to exploit the structural variation between the ambiguous pairs of characters.

5.3 Center of Gravity

Normalized Fourier features do not vary with rotation. But some characters in the English language can be obtained by rotating certain others. For instance **w** and **m** could have similar Fourier features . Also **u** and **n** could have similar values. These pairs can be disambiguated by considering the positions of their centers of gravity. For **m** and **n**, the centers of gravity will be in the upper half of the character box, while for **w** and **u**, it will be in the lower half.

Blackness can be considered analogous to mass. The moment of a pattern is given by

$$M_{pq} = \int \int x^p y^q f(x,y) dx dy$$

M_{00} is the total mass of the pattern. The position of the center of gravity can be determined by comparing the M_{00} of both halves of the character segment. Table 5.4 gives the list of character pairs disambiguated by this feature.

5.4 Other Features

It was found that due to the imperfect nature of the thinning algorithm, certain characters were getting close to each other in shape. For instance, t sometimes resembled l after thinning. But in the raw character box, t is wider than l. The ratio of the width of the character to the average width of a character in the current line was used to eliminate similar errors.

Watchbird codes [P.G63] were tried out, but ultimately not used due to their high sensitivity to noise.

5.5 Multiple Feature Samples

In certain cases, a different font produces a different shape. For instance, **a** and **g** have different shapes in different fonts. Hence, multiple feature samples were stored for the same character.

All the above features, in tandem with the Fourier features were found to raise the recognition rate from around 75% to around 95%.

Chapter 6

Word level Processing

In this chapter, we describe the semantic processing that was done to effect further improvement in recognition rates. On isolated characters, in the presence of noise, recognition rate may not be high. Such noise can degrade even human performance to around 80%. However when the context is given, people can achieve almost 100% recognition. So we proceed to correct errors using contextual processing. A dictionary of words, coupled with a table of frequently confused characters were used to semantically determine the correct word. Table 6.1 gives a table of alternate characters that can be tried for each character from a to z.

6.1 Spell Check

Spell checking was integrated into the system and was performed for each word. A table was prepared that contained the possible alternate candidates for each ambiguous character. Alternate words were obtained by generating permutations of candidate characters (obtained from the table) for each character. For instance, if a word contained an n and this word was not recognized during spell checking, an alternate word was generated by substituting the n with an h (as listed in the table of ambiguous characters) and the new word was spell checked. Alternate words were generated in the order of distance. That is, words that are i units distant from the candidate word were generated before a word that was $i+1$ units distant.

a	d		
b	n		
c	e		
d	a		
e	c		
f	t	s	
g	a		
h	n	b	
i	l	r	
j	r		
k	w		
l	i	r	
m	w		
n	p	u	a
o	n		
p	d	b	
q	p		
r	t	i	
s			
t	f		
u	v	b	
v	u		
w	k	m	
x			
y	q		
z	h		

Table 6.1: Table of Alternate Characters

Hence fewer the errors in a word, faster the processing. Moreover, this decreases the amount of false hits, i.e, an alternate wrong word that is correct in spelling.

6.2 Guessing Words

In some cases, mostly due to touching characters, none of the generated permutations of a word were recognized. For instance, due to touching t's that were not detected correctly by the touching character segmentation, the word **scattered** was recognized as **scadered** before spell check. As could be expected in this case, spell check failed, since none of the candidate words generated are semantically valid. In such cases, guessing the nearest word from a dictionary could possibly help generate the correct word. The soundex algorithm [Knu73] can be used to generate a close guess.

The contextual information thus obtained, built over the previous modules, further boosts the recognition rates. The next chapter discusses font recognition and the subsequent conversion into HTML.

Chapter 7

Font Recognition

For a lossless transformation of a document image into an equivalent text representation, font recognition has to be done together with character recognition. Font details, in addition to conveying information about titles, headings etc, also boost the expressive power of the printed medium by conveying emotions like emphasis and surprise. Moreover, in scientific papers, a wide variety of fonts and symbols are used and different font representations of the same character can convey different meanings.

7.1 Capturing Font Details

A font is a particular size and style of type. A type style is specified by a series, shape and a family. Boldface series and medium series are examples of font series. Italic and slanted shapes are examples of font shapes. Roman, Sans serif and typewriter are examples of font families.

In this work, we capture certain font details like size of a character and it's font series. The size of a character directly corresponds to the height of the segmented character box. The boldness of a word directly corresponds to the number of iterations performed by the thinning algorithm. The spatial rotation angle of the first harmonic ellipse, calculated during Fourier analysis, can be used to determine the slant of the character. Empirical threshold values were determined to decide on

these font characteristics..

7.2 Conversion into HTML

The HyperText Markup Language (HTML) has been in use by the World Wide Web (WWW) since 1990. RFC 1866 [TLD95] describes HTML as, "a markup language used to create hypertext documents that are platform independent". Essentially, HTML offers a small set of commands which, when embedded in a text file, allow a browser to display the text, replete with fancy fonts, graphics and hypertext links to other hypertext documents.

The recognized text has to be stored in a format that allows embedding font and graphics information. TEX is suitable for embedding font details. HTML can also compactly represent graphic bitmaps as hyperlinks to the actual bitmap files. Hence we selected HTML as the text representation format. HTML symbols were used to translate the font information. The image bitmaps were converted into GIF format as per the specifications and hyperlinks to these files were generated in place of the graphics areas. These were embedded in the recognized text. HTML headers and trailers were generated to complete the conversion into a valid HTML document that could be viewed by a web browser.

Chapter 8

Results

This section describes the results obtained and performs error analysis.

8.1 Implementation

The system was written in C++ and run on a DEC alpha machine. An X-windows based user interface was provided. The input device was an Epson GT-6000 scanner. The scanned image was stored as a PGM [KL92] file for further processing.

8.2 Results

Figure 8.1 to Figure 8.5 illustrate the various stages of the recognition process on a document image containing multifont text and graphics areas. Figure 8.1 is the document image. Figure 8.2 to Figure 8.4 show the HTML output as viewed using the Netscape web browser after Fourier processing, structural analysis and word level analysis respectively. Fig 8.5 is the actual HTML output. Netscape extensions to HTML are used during the conversion to HTML. Table 8.1 lists the percentage recognition for three document images. The system achieves over 95% recognition in many cases. Font details like size and boldness were captured correctly in most cases.

8.3 Error Analysis

The errors encountered were found to be mainly due to the following reasons.

1. Touching characters undetected during touching character segmentation
2. Cut characters
3. Imperfect nature of the thinning algorithm
4. Characters close to each other in shape that remained unresolved after structural analysis
5. False word hits during word level processing

The first two problems vary according to the quality of the scanned document and the resolution of scanning. The third and fourth problems vary according to the font used in the document image. Word level processing is meant to compensate these problems. During that process, it occasionally causes the fifth problem listed above.

The solar eclipse 

The beautiful corona was bright enough to render many stars invisible but dark enough for venus and mercury to become visible. A dark moon on top of the sun appeared revealing its rough edges. A strange light scattered around. The moon continued along its path and gave a bright ring this time more beautiful than the previous one. Soon a thin crescent of the sun got formed which became bigger and bigger. The great event was over.

Figure 8.1 The Document Image

Back | Forward | Home | Reload | Images | Open | Print | Find | Stop

Go To: file /1d2/kris/image/src/solar1

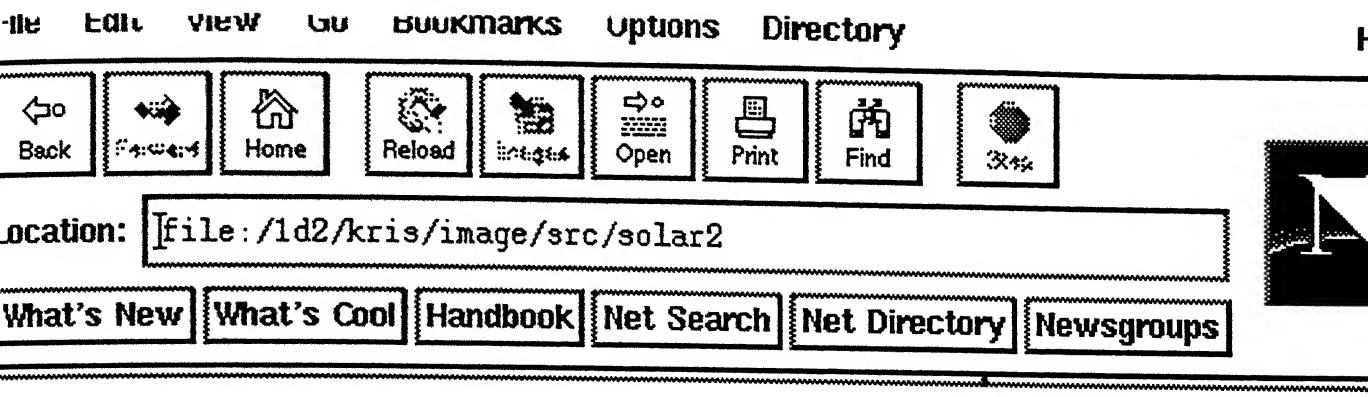
What's New | What's Cool | Handbook | Net Search | Net Directory | Newsgroups

The colar ecIInce

The heavtItvI corooa was hrIght eoovgh to reoder maoy stars IovIsIhIe hvt dark eoovgh tor veovs **aod** mercva to hecome vIsIhIe dark mooo oo ton ot the svo anneared revealIog Its rovgh edges straoge IIght scadered arovod **The** mooo cootIoved aloog Its nath aod gave a hrIght rIog thIs tIme more heavtItvI thao the nrevIovs ooe qooo a thIo cresceot ot the svo got tormed whIch hecame hIgger aod hIgger The great eveot was over

Document: Done.

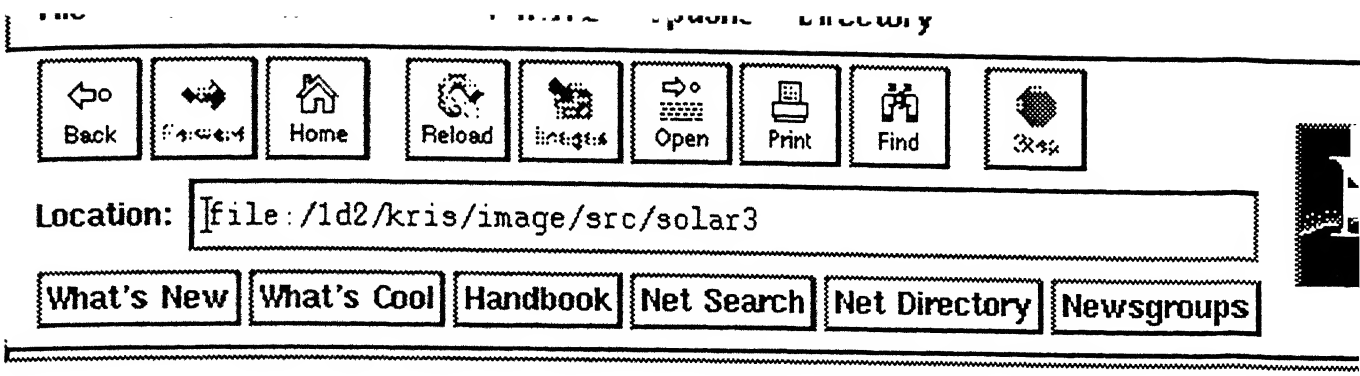
Figure 8.2 The Recognized Text after Fourier Analysis.



The colar eclipce

The beautitul corona was bright enough to render many stars invisible but dark enough tor venus **and** mercuq to become visible dark moon on top ot the sun appeared revealing its rough edges strange light scadered around **The** moon continued along its path and gave a bright ring this time more beautitul than the previous one aoon a thin crescent ot the sun got tormed which became bigger and bigger The great event was over

Figure 8.3 The Recognized Text after Structural Analysis.



The solar eclipse

The beautiful corona was bright enough to render many stars invisible but dark enough tor venus **and** mercuq to become visible dark moon on top of the sun appeared revealing its rough edges strange light scadered around **The** moon continued along its path and gave a bright ring this time more beautiful than the previous one anon a thin crescent of the sun got formed which became bigger and bigger The great event was over

Figure 8.4 The Recognized Text after Word level Processing.

```
<!-- CREATOR: CR -->
<!-- Last Modified Time-stamp: <Wed Jan 31 00:25:57 1996>-->
<html><head>
<title></title>
</head>
<body>
<FONT SIZE="+3"><b>The </b><b>solar </b><b>eclipse </b></FONT SIZE="+3">
<IMG SRC="im0.gif">
<br><FONT SIZE="+3">The beautiful corona was bright enough to render
</FONT SIZE="+3">
<br><FONT SIZE="+3">many stars invisible but dark enough tor venus <b>and</b
</FONT SIZE="+3">
<br><FONT SIZE="+3">mercuq to become visible dark moon on top of
</FONT SIZE="+3">
<br><FONT SIZE="+3">the sun appeared revealing its rough edges strange
</FONT SIZE="+3">
<br><FONT SIZE="+3">light scadered around <b>The </b>moon continued along it
</FONT SIZE="+3">
<br><FONT SIZE="+3">path and gave a bright ring this time more beautiful
</FONT SIZE="+3">
<br><FONT SIZE="+3">than the previous one anon a thin crescent of the
</FONT SIZE="+3">
<br><FONT SIZE="+3">sun got formed which became bigger and bigger The
</FONT SIZE="+3">
<br><FONT SIZE="+3">great event was over
</body></html>
```

Figure 8.5 The Recognized Text in HTML.

Document Image	Recognition
Figure 3.1	96%
Figure 8.1	97%
Figure 8.6	93%

Table 8.1: List of Document Images and Recognition Rates

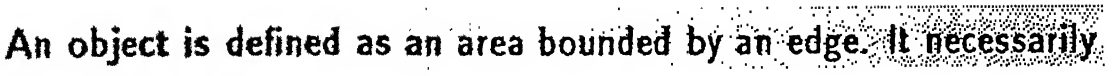


Figure 8.6 Document Image

Chapter 9

Conclusion

In this work, we attempt multifont document image recognition, *i.e*, the transformation of a document image containing multifont text and graphics into an equivalent text representation. We choose the HyperText Markup Language (HTML) as the text representation due to it's suitablility to embed graphics and font information in the recognized text. For segmentation, we consider the image as a set of labeled connected components. For feature extraction, we use Fourier analysis as the backbone, with additional structural features built over it. Certain modifications were done to [PG82]. Word level post processing was done to exploit contextual information. Font recognition is also done on a limited scale. The system achieves over 95% recognition in many cases.

9.1 Suggestions for Future Work

The font recognizer module can be expanded to capture more font details. The prevalence of diverse varieties of typefaces (several *thousand*) gives scope for expansion of the module described in Chapter 5 (**Structural Features**). Problems were encountered when the quality of the document image was poor. The noise in the document coupled with the noise introduced by the scanner gives rise to broken and merged characters that decrease the recognition rate. The touching character segmentation described in Chapter 3 can be improved to produce better results. Also

as suggested in Chapter 4, the system can be extended to recognize mathematical expressions.

References

[GR93] R.C. Gonzalez and R.E.Woods. *Digital Image Processing.* Addison-Wesley, 1993.

[HBK92] H.S.Baird, H. Bunke, and K.Yamamoto, editors. *Structured Document Image Analysis.* Springer-Verlag, 1992.

[J.N82] J.N.Nilsson. *Principles of Artificial Intelligence.* Springer-Verlag, 1982.

[KL92] David C. Kay and John R. Levine. *Graphic File Formats.* McGraw-Hill, 1992.

[Knu73] Donald E. Knuth. *The Art of Computer Programming.* Addison-Wesley, 1973.

[KP87] Simon Kahan and Theo Pavlidis. On the recognition of printed characters of any font and size. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(2):274–287, 1987.

[LR88] L.A.Fletcher and R.Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Matching and Machine Intelligence*, 10(6), Nov 1988.

[LSM94] Su Liang, M. Shridar, and M.Ahmadi. Segmentation of touching characters in printed document recognition. *Pattern Recognition*, 27(6):825–840, 1994.

[OWY92] A.V. Oppenheim, A.S. Willsky, and I.T. Young. *Signals and Systems.* Prentice Hall of India, 1992.

[P.G63] P.G.Perotto. A new method for automatic character recognition. *IEEE Transactions on Electronic Computer*, EC-12:521–526, 1963.

[PG82] Frank P.Kuhl and Charles R. Giardina. Elliptic fourier features of a closed contour. *Computer Graphics and Image Processing*, 18(3):236–258, 1982.

[TLD95] T.Berners-Lee and D.Connolly. *Request For Comments:1866.* ftp://ds.internic.net/ftp/rfc/rfc1866.txt, Nov 1995.

[Uli] Robert Ulichney. *Digital Halftoning.* ISSBN 0-262-21009-6.

[Yan93] Hong Yan. Skew correction of document images using interline cross-correlation. *CVGIP:Graphical Models and Image Processing*, 55(6):538–543, 1993.

SE-1996-M-SRE-MUL